	SISTEMAS OPERATIVOS. CUESTIONES **10 de Septiembre de 2014.**
	Nombre_____DNI_____ Apellidos_____ Grupo_____

Cuestión 1. (0,75 puntos) Sistema de Ficheros Un dispositivo de memoria flash de 1 GB de capacidad, direcciones de 32 bits y bloques de 128 KB, contiene un sistema de ficheros FAT.

a. ¿Cuántos bytes son necesarios para almacenar la tabla FAT?
b. Si un archivo tiene 1MB indique cuántos bloques ocupa y cuántos accesos a disco son necesarios para acceder a uno de sus bloques en el caso mejor y en el caso peor. Asumir que no hay ningún dato relacionado con el sistema de ficheros en memoria *RAM,* que el fichero se encuentra en el directorio raíz y que el directorio cabe en un bloque.

Cuestión 2. (0,75 puntos) E/S En la práctica de E/S se proponen las siguientes funciones para implementar las operaciones open y release del módulo:

```
static int
device_open(struct inode *inode,
            struct file  *file) {
  static int counter = 0;
  if (Device_Open)
    return -EBUSY;
  Device_Open++;
  sprintf(msg, "bla,bla %d", counter++);
  msg_Ptr = msg;
  try_module_get(THIS_MODULE);
  return SUCCESS;
}
```

```
static int
device_release(struct inode *inode,
               struct file  *file) {
  Device_Open--;
  module_put(THIS_MODULE);
  return 0;
}
```

Explica cuál es la función de las funciones *try_module_get* y *module_put*. Indica qué comando Linux puede utilizarse para saber cuántos procesos tienen un módulo en uso.

Cuestión 3. (0,75 puntos) E/S Un determinado módulo Linux se inicializa del siguiente modo:

```
#define MAJOR 10
static struct
file_operations fops = {
  .read = device_read,
  .write = device_write,
  .open = device_open,
  .release = device_release
};
```

```
int
init_module(void) {
  Major = register_chrdev(MAJOR, DEVICE_NAME, &fops);
  if (Major != MAJOR ) {
    printk(KERN_ALERT"Registering failed for %d\n", MAJOR);
    return Major;
  }
        ...
}
```

Para trabajar con el módulo se crea un fichero de dispositivo con el siguiente comando: *`mknod /dev/disp0 c 10 0`*. Explica el significado de cada uno de los parámetros de este comando. Indica cuál o cuáles de las funciones registradas por *register_chrdev* se ejecutarán cuando el usuario ejecute el siguiente comando: *`cat /dev/disp0`*.

Cuestión 5. (0,75 puntos) Memoria En un sistema con memoria virtual paginada explicar cómo funciona el algoritmo LRU indicando la información que almacena para cada marco de página y cómo la utiliza, número de fallos de página, … Usar como ejemplo una memoria física de cuatro marcos de página (inicialmente los marcos de página están vacíos) y la siguiente cadena de referencias: a d c a b d c a e f a e d.

Cuestión 6. (0,75 puntos). Procesos/hilos Considere el siguiente código:

```
int fd = -1;
char buf1[4]=”aaaa”;
char buf2[4]=”bbbb”;

int main() {
  pthread_t tid1, tid2;
  pthread_create(&tid1,NULL,h1,NULL);
  pthread_create(&tid2,NULL,h2,NULL);
  pthread_join(tid1,NULL);
  pthread_join(tid2,NULL);
  close(fd);
}
```

```
void* h1( ) {
  fd=open("prueba",O_RDWR|O_CREAT|O_TRUNC,0666);
  write(fd,buf1,4);
}

void* h2( ) {
  while (fd==-1) {};
  write(fd,buf2,4);
}
```

Indique si cada una de las siguientes afirmaciones es cierta o falsa justificando la respuesta (cualquier respuesta sin justificar será considerada errónea):

a. El hilo 2 (función *h2)* nunca saldrá del bucle *while* inicial.
b. La escritura del hilo 2 (función h2) producirá un error por no haber abierto antes el fichero.
c. El contenido final del fichero *prueba* será o bien, “aaaa” o bien “bbbb”; ninguna otra alternativa es posible.
d. La llamada a *close()* del programa principal no debería devolver ‘-1’ (esto es, no debería producir ningún error).

Cuestión 7. (0,75 puntos). Procesos/Hilos. Estudie el siguiente código e indique qué se mostrará por pantalla tras su ejecución. Si existe más de un posible orden de ejecución (y por tanto, más de una posible salida), indique todas las posibilidades. Justifique su respuesta.

```
int a=1;
sem_t s;
int main() {
  pthread_t tid1, tid2;
  int b = 1;
  sem_init(&s,0,0);
  for (int i=0; i < 3; i++) {
    pthread_create(&tid1,NULL,h1,b);
    pthread_create(&tid2,NULL,h2,b);
    b++;
    pthread_join(tid1,NULL);
    pthread_join(tid2,NULL);
  }
  printf("Main: a=%d, b=%d\n", a,b);
}
```

```
void* h1(int x) {
  sem_wait(&s);
  a = a +x;
  printf("H1: a=%d, x=%d\n",a,x);
  return 0;
}

void* h2(int x) {
  a = a * x;
  printf("H2: a=%d, x=%d\n",a,x);
  sem_post(&s);
  return 0;
}
```

N.B. Se ha simplificado el paso de parámetros a un hilo para facilitar la comprensión del código. La interfaz POSIX especifica que ese argumento debe ser un puntero, pero para este ejercicio asumimos que es posible pasar directamente un entero.

Cuestión 8. (0,75 puntos) Sincronización. La función de bloqueo en una variable de condición tiene el siguiente prototipo:

	SISTEMAS OPERATIVOS. PROBLEMAS **10 de Septiembre de 2014.**
	Nombre_____DNI_____ Apellidos_____ Grupo_____

Problema 1. (2 puntos) Un sistema de ficheros de un SO diseñado a partir de UNIX utiliza bloques de disco de 16 bytes de capacidad. Para el direccionamiento de estos bloques se utilizan punteros de 8 bits. Cada nodo-i tiene 1 puntero de direccionamiento directo y 1 puntero indirecto simple. Parte del contenido del sistema de ficheros está indicado en las siguientes tablas:

Mapa de bits (de bloques de datos. '1' es ocupado. El primer índice se corresponde con el bloque 0): 00010111010000010110...0

Tabla de nodos-i (el nodo-i 2 es la raíz del árbol)

nodo-i	2	3	4	6	7	8	9	10
Enlaces	NA	NA	NA	NA	1	1	1	...
Tipo F/D	D	D	D	D	F	F	F	...
Directo	3	5	6	7	9	12	14	...
Indirecto								

Lista de bloques:

Bloque 3

.	2
..	2
opt	3
tmp	4

Bloque 5

.	3
..	2
var	6

Bloque 6

.	4
..	2

Bloque 7

.	6
..	3
man	9
ell	7

Bloque 9

He_who_controls_

a. Dibuje el árbol de directorios empleando óvalos para los directorios y rectángulos para los ficheros (la raíz del árbol es el nodo-i 2)
b. Modifique el sistema de ficheros (tablas de nodos-i, lista de bloques y mapa de bits) para que el fichero cuyo primer bloque de datos está en el bloque 9 contenga la frase "*He_who_controls_the_past_controls_the_future*" (recordad que cada bloque almacena 16 bytes y que cada carácter ocupa un byte).
c. Un usuario crea un enlace físico llamado "*erw*" al fichero "*ell*" en el directorio "tmp". ¿Cómo

Problema 2. (2 puntos) Simular con hilos el comportamiento de una gasolinera con 2 surtidores de pago directo con tarjeta. Cuando un cliente coge un surtidor puede servirse el combustible deseado (con una llamada a la función `void ServirCombustible( int surtidor, int dinero )`). Una vez servido el combustible dejará libre el surtidor para que otro cliente pueda usarlo. Para que funcione correctamente el servicio deben satisfacerse las siguientes restricciones:

- Los clientes deben acceder al servicio en orden de llegada, pudiendo escoger cualquiera de los surtidores libres.
- No puede haber más de un cliente simultáneamente en un mismo surtidor.
- Mientras un cliente se está sirviendo el combustible en un surtidor, cualquier otro cliente debe poder utilizar el otro surtidor si está libre.

Implementar el código de la función principal del hilo cliente: `void cliente(int dinero).` El argumento de entrada de dicha función indica cuánto dinero invertirá ese cliente en el combustible.

```
< declaración de variables globales >

void cliente(int dinero){

        < declaración de variables locales >

        // Comprobar que se cumplen los requisitos para repostar

        ServirCombustible(surtidor,dinero);

        // Acciones de salida

}
```